Dynamic Multiple Parity (DMP) Disk Array for Serial Transaction Processing

K.H. Yeung, *Member*, *IEEE*, and T.S. Yum, *Senior Member*, *IEEE*

Abstract—The performance of today's database systems is usually limited by the speed of their I/O devices. Fast I/O systems can be built from an array of low cost disks working in parallel. This kind of disk architecture is called RAID (Redundant Arrays of Inexpensive Disks). RAID promises improvement over SLED (Single Large Expensive Disks) in performance, reliability, power consumption, and scalability. However, a general fact about RAID is that the "write" operation is difficult to speedup. In this paper, we propose a new RAID architecture, called **Dynamic Multiple Parity (DMP) Disk Array**, for serial transaction processing database systems. Serial transaction processing database systems include engineering database systems, fully replicated database systems using a completely centralized algorithm and distributed systems using the conservative timestamp ordering algorithm. DMP Disk Array can significantly increase the I/O throughput by incorporating multiple parity disks. Due to the inherent distributed sparing property, DMP Disk Array can provide normal service to the users under single disk failure condition. Delay and maximum throughput analysis on DMP Disk Array is performed. Results show that, for a typical "write" job proportion of 20 percent, DMP Disk Array can provide nearly 20 percent improvement on I/O throughput over that of RAID level 5 when one extra parity disk is used.

Index Terms—RAID, disk arrays, I/O systems, database systems, transaction processing.

1 Introduction

In the past decade, considerable attention has been drawn to the research and development of database computers. As Su [1] has stated in his book, there are three reasons why database computers are needed: 1) the need for efficient and effective data management, 2) the need for more powerful database management systems, and 3) high performance database computers become economically feasible because of the advancement in hardware technology and the reduction in hardware cost. The performance of a database system is usually limited by the speed of its storage devices. One good example is the Datacycle™ project at BellCore starting in the late 1980s [2]. BellCore proposed an innovative database architecture called the Datacycle™. In this architecture, the entire database is periodically pumped out from the central database to a number of servers in which the user required data are filtered out. Since the whole database is pumped out, Datacycle™ has an unlimited throughput for read-only transactions. Datacycle™ technique assumes that the database is "memory resident," i.e., the entire database has to reside in very fast storage.[1] This assumption, however, limits its scope of applications.

Redundant Arrays of Inexpensive Disks (RAID) is an innovative concept in designing fast and reliable data storage systems. The philosophy behind RAID is that, instead of using one single expensive disk to achieve the performance and reliability required, an array of low cost disks working in parallel are used. Five levels of RAID were defined when RAID was first introduced[2] and RAID level 5 was found to be one of the best [3]. For all levels of RAID, the "write" operations are much slower than the "read" ones. This limitation is particularly severe for applications with frequent data updates.

There are two reasons why a "write" operation takes more time for RAID. First, a "write" operation involves the additional step of reading back the old data from one disk and parity from another disk. Second, a "write" operation involves the waiting time for two specific disks to be free simultaneously before actual writing. This waiting time can be reduced by using the technique presented in this paper.

In this paper, we propose a new RAID architecture, called Dynamic Multiple Parity (DMP) Disk Array, for serial transaction processing database systems. Many database systems process transactions in this way. Examples are engineering database systems [4], [5], [6], fully replicated database systems using the completely centralized algorithm, and distributed systems using the conservative timestamp ordering algorithm [7], [8], [9]. When we discuss the operation of DMP Disk Arrays, we will see how this kind of database systems handles I/O requests in a way different from other database systems. We will also show that DMP Disk Array can significantly reduce the waiting time of disk operations and provide a higher I/O throughput than RAID level 5. In the next section, we describe DMP Disk Array in detail. We then present an average delay analysis in Section 3 and a maximum throughput analysis in Section 4. In Section 5, results of simulation with a precise disk model are given. We then conclude the paper in Section 6.

1. In the prototype built by BellCore, the whole database resides in RAM.

2. Some disk manufactors introduced their own levels of RAID later.

• *K.H. Yeung is with the Department of Electronic Engineering, City University of Hong Kong, Yau Yat Chuen, Hong Kong.*
E-mail: eeayeung@cityu.edu.hk.
• *T.S. Yum is with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, Hong Kong.*
E-mail: yum@ie.cuhk.edu.hk.

Manuscript received 2 Aug. 1994; accepted 26 Feb. 2001.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 113686.

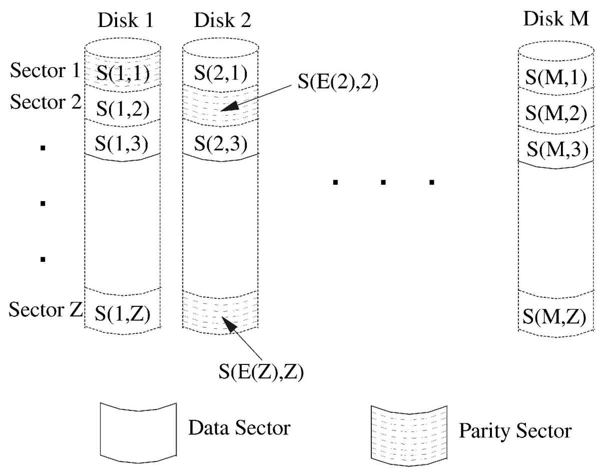

Fig. 1. The sector coordinate system showing level 5 RAID.

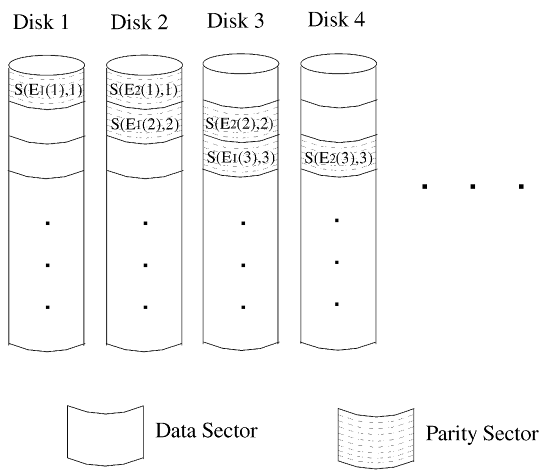

Fig. 2. The placement of parity sectors for DMP Disk Array when $R = 2$.

2 DMP Disk Array

2.1 Sector Coordinate System

The sector coordinate system can be used to describe RAID operations, including that of DMP Disk Array. Consider a disk array system with M disks, where each disk consists of Z sectors and each sector can store K bits of information (Fig. 1). Let $S(i,j) = (b_1^{i,j}, b_2^{i,j}, b_3^{i,j}, \dots, b_K^{i,j})$ be the bit pattern of sector j in disk i. It can be data or parity. As an example, a RAID level 1 is described in this sector coordinate system as:

$$S(i,j) = S(i+1,j) \;\; j = 1,2,3,\dots,Z, \; i = 1,3,5,\dots,M-1. \tag{1}$$

Similarly, RAID level 4 and level 5 can be described as:

$$\sum_{i=1}^{M} S(i,j) = \begin{cases} (1,1,1,\dots,1) & \text{for all } j \quad \{\text{odd parity}\} \\ (0,0,0,\dots,0) & \text{for all } j \quad \{\text{even parity}\}, \end{cases} \tag{2}$$

where Σ is defined here as the mod-2 sum of the sectors' contents:

$$\begin{aligned} \sum_{i=1}^{M} S(1,j) &= S(i,j) \oplus S(2,j) \oplus \dots \oplus S(M,j) \\ &= (b_1^{1,j}, b_2^{1,j}, \dots, b_K^{1,j}) \oplus (b_1^{2,j}, b_2^{2,j}, \dots, b_K^{2,j}) \oplus \dots \\ &\quad \oplus (b_1^{M,j}, b_2^{M,j}, \dots, b_K^{M,j}) \\ &= (b_1^{1,j} \oplus b_1^{2,j} \oplus \dots \oplus b_1^{M,j}, b_2^{1,j} \oplus b_2^{2,j} \oplus \dots \oplus b_2^{M,j}, \dots, \\ &\quad b_K^{1,j} \oplus b_K^{2,j} \oplus \dots \oplus b_K^{M,j}). \end{aligned} \tag{3}$$

Let $E(j)$ be the location of the parity sector at row j. For RAID level 4 $E(j) = M$ and for RAID level 5 $E(j) = [(j-1) \bmod M] + 1$. The placement of these parity sectors is shown in Fig. 1. A study of the various parity placement methods for RAID level 5 can be found in [10].

2.2 Sector Organization

The DMP Disk Array proposed in this paper is a new RAID architecture for which RAID level 5 is a special case. RAID level 5 has only one parity sector in each sector row and, therefore, it is not possible to simultaneously update two or more sectors on the same row. DMP Disk Array allows such updates by placing R parity sectors in each row. Their locations $E_1(j), E_2(j), \dots, E_R(j)$ for row j are

$$E_r(j) = [(r + j - 2) \bmod M] + 1 \qquad r = 1,2,\dots,R. \tag{4}$$

Lee and Katz [10] showed that, for relatively large request sizes of hundreds of kilobytes, the choice of parity placement can significantly affect the performance of disk arrays, whereas, for small request sizes, the choice of parity placement is insignificant to system performance. We therefore arbitrarily choose the parity locations as stated in (4). Although there are R parity sectors in each row, parity integrity described in (2) is always maintained for DMP Disk Array. (An example on the sector organization of DMP Disk Array for $R = 2$ is shown in Fig. 2.)

There are two advantages to placing R parity sectors in each row. First, for each data sector modification, we can choose any one of the R parity sectors in the same row for simultaneous parity modification. Hence, blocking due to busy disks can be significantly reduced. Second, up to R data sectors on the same row can now be modified simultaneously. We now prove that DMP Disk Array has these useful properties.

2.3 Properties of DMP Disk Array

Property 1 concerns the simultaneous negation of two bits on the same row.

Property 1. *For any row j, simultaneous negation of any two bits in the same bit positions of two different sectors will not affect the parity sum of the row.*

Proof. Consider the set of rth bits of each sector in sector row j, i.e., $b_r^{1,j}$, $b_r^{2,j}$, ..., and $b_r^{M,j}$. In order to maintain parity integrity, the sum of these bits should always be 1 for odd parity or 0 for even parity. The parity sum after the negation of any two bits $b_r^{a,j}$ and $b_r^{b,j}$, where $a \neq b$ is:

$$\begin{aligned}
&b_r^{1,j} \oplus \ldots \oplus \overline{b_r^{a,j}} \oplus \ldots \oplus \overline{b_r^{b,j}} \oplus \ldots \oplus \overline{b_r^{M,j}} \\
&= \overline{b_r^{a,j}} \oplus \overline{b_r^{b,j}} \oplus b_r^{1,j} \oplus \ldots \oplus b_r^{M,j} \\
&\qquad \{\text{commutative law}\} \\
&= \left(\overline{b_r^{a,j}} \oplus \overline{b_r^{b,j}}\right) \oplus \left(b_r^{1,j} \oplus \ldots \oplus b_r^{M,j}\right) \\
&\qquad \{\text{associative law}\} \\
&= \left(\overline{\overline{b_r^{a,j}}}\, b_r^{b,j} + \overline{b_r^{a,j}}\, \overline{\overline{b_r^{b,j}}}\right) \oplus \left(b_r^{1,j} \oplus \ldots \oplus b_r^{M,j}\right) \\
&\qquad \{\text{definition of exclusive-or}\} \qquad\qquad (5) \\
&= \left(b_r^{a,j}\overline{b_r^{b,j}} + \overline{b_r^{a,j}}\, b_r^{b,j}\right) \oplus \left(b_r^{1,j} \oplus \ldots \oplus b_r^{M,j}\right) \\
&\qquad \{\text{law of double negation}\} \\
&= \left(b_r^{a,j} \oplus b_r^{b,j}\right) \oplus \left(b_r^{1,j} \oplus \ldots \oplus b_r^{M,j}\right) \\
&\qquad \{\text{definition of exclusive-or}\} \\
&= b_r^{a,j} \oplus b_r^{b,j} \oplus b_r^{1,j} \oplus \ldots \oplus b_r^{M,j} \\
&\qquad \{\text{associative law}\} \\
&= b_r^{1,j} \oplus \ldots \oplus b_r^{a,j} \oplus \ldots \oplus b_r^{b,j} \oplus \ldots \oplus b_r^{M,j} \\
&\qquad \{\text{commutative law}\}.
\end{aligned}$$

□

The last result is identical to the parity sum before the simultaneous negation. We can extend the argument on the modification of two bits to that of the modification of two sectors on the same row. This is stated as Property 2.

Property 2. *Parity integrity of a row can be maintained by modifying any one of the parity sectors in the same row.*

Property 2 implies that there can be R different ways to update a data sector. Due to this flexibility, the probability that a "write" request is blocked due to the busy disk can be reduced.

Property 3. *Consider the simultaneous modification of data in sector j of disk a and parity in sector j of disk b. We denote the old data sector, the new data sector, the old parity sector, and the new parity sector as $S(a,j)|_{old}$, $S(a,j)|_{new}$, $S(b,j)|_{old}$, and $S(b,j)|_{new}$, respectively. For maintaining data integrity, the new parity sector should be:*

$$S(b,j)|_{new} = S(a,j)|_{old} \oplus S(a,j)|_{new} \oplus S(b,j)|_{old}. \qquad (6)$$

Proof. In order to maintain parity integrity, the partial sum of the two sectors $S(a,j)$ and $S(b,j)$ must not be changed after sector modification, i.e.,

$$S(a,j)|_{new} \oplus S(b,j)|_{new} = S(a,j)|_{old} \oplus S(b,j)|_{old}. \qquad (7)$$

Solving for $S(b,j)|_{new}$, (6) is obtained. □

Property 4. *For DMP Disk Array with R parity sectors in each row, R data sectors located in the same row can be simultaneously updated.*

Proof. Equation (7) shows that, for any sector update on row *j*, the partial sum of the data sector and the parity sector will always be the same. Thus, the particular sector update will not affect the updating of the other sectors in row *j*. From Property 3, we find that each sector update requires the old contents of two sectors only. Therefore, each update is actually carried out by two disks working in cooperation and is independent of the operations of the rest of the disks. Therefore, with R parity sectors, R simultaneous updates can be performed. □

Property 5. *Consider a DMP Disk Array with M disks and R parity sectors in each row ($R > 1$). When a disk fails, it is possible to reconfigure the remaining $M-1$ disks to a new array with $R-1$ parity sectors in each row without data loss.*

Proof. Let (p, p, ..., p) be the parity sum of all the sectors in a row, say row *j*, and let $S(b_1,j), S(b_2,j), \ldots, S(b_R,j)$ be the parity sectors. Obviously,

$$b_1 = E_1(j), b_2 = E_2(j), \ldots, b_R = E_R(j).$$

Suppose disk i fails. We consider two cases for the recovery of $S(i,j)$.

1. If $S(i,j)$ is a data sector, it can be recovered from

$$S(i,j) = (p,p,\ldots,p) \oplus \sum_{n=1,n\neq i}^{M} S(n,j). \qquad (8)$$

One of the parity sectors, say $S(b_R,j)$, can be used to store the recovered data $S(i,j)$. To maintain parity integrity another parity sector, say $S(b_1,j)$, is modified as:

$$S(b_1,j) = S(b_1,j) \oplus S(b_R,j). \qquad (9)$$

The recovered DMP Disk Array now has $R-1$ parity sectors.

2. If $S(i,j)$ is a parity sector, then no data is lost. The parity integrity can be recovered by modifying another parity sector, say *b*, as follows:

$$S(b,j) = (p,p,\ldots,p) \oplus \sum_{\substack{n=1,n\neq i,\\ n\neq b}}^{M} S(n,j). \qquad (10)$$

□

Property 5 tell us that DMP Disk Array has the distributed sparing property discussed in [11]. It is shown in [11] that distributed sparing is the best sparing technique for small disk arrays.

2.4 Principle of Operation

Fig. 3 shows the organization of DMP Disk Array. Requests from a host are directly sent to the disk controller for the I/O operations. The disk controller consists of four parts: an FCFS queue, a local memory, a scheduling processor, and a DMA controller. The queue is used for storing I/O requests. Since we are considering systems which execute transactions in a strict order, a single global queue with FCFS service discipline is used. Note that this is different from other systems reported in the literature, which use separate disk queues [12], [13], [14], [15]. The data associated with each request (i.e., the new data for a sector) is stored in the local memory when the request is placed on the queue. The local memory is also used for buffering the data sent to/read from each of the disks

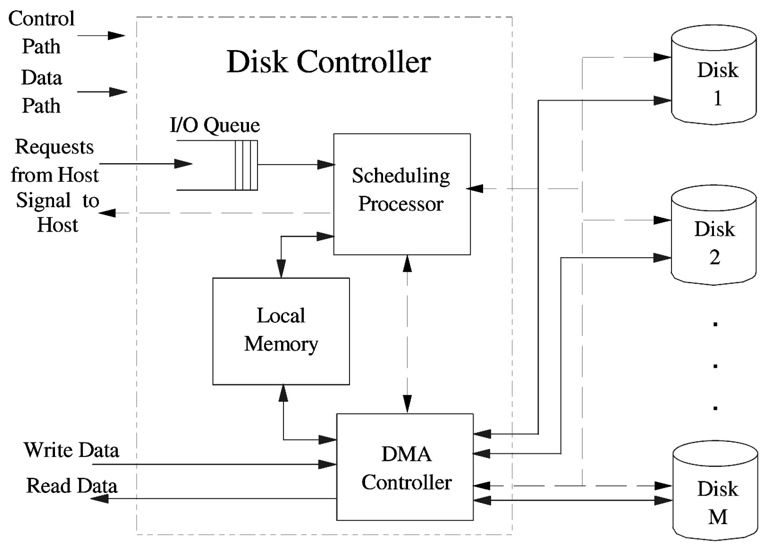

Fig. 3. Organization of DMP Disk Array.

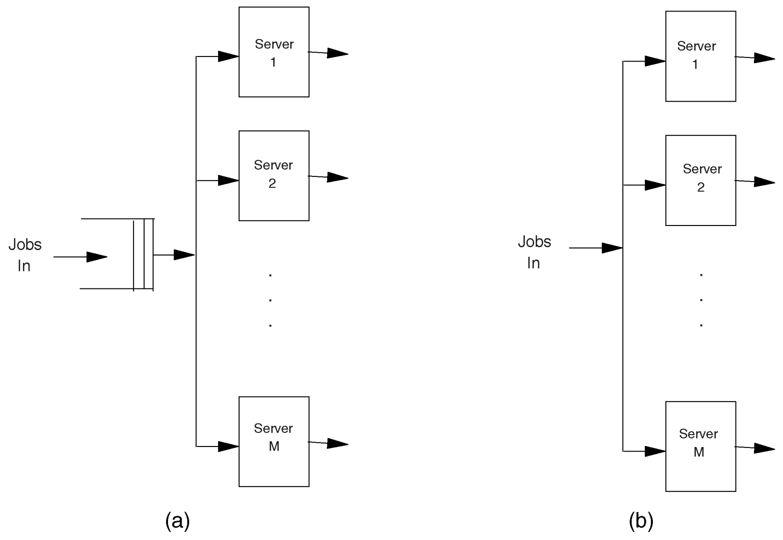

Fig. 4. Queuing models of DMP Disk Array.

with the help of the DMA controller. The DMA controller functions basically as a multiplexer/demultiplexer and handles simultaneous data transfers to various disks. The scheduling processor is responsible for distributing I/O requests to the disks and performs all necessary processing. Specifically, its functions are outlined as follows:

1. It reads a request from the FCFS queue when ready and determines whether the request is a "read" or a "write" type.
2. For a "read" request, the processor will
   a. check the status of the disk involved with this request;
   b. instruct the disk involved to read the target sector;
   c. load the sector to the local memory; and
   d. signal the host for data ready.
3. For a "write" request, the processor will
   a. check the status of the disks and select at random the parity sector of a nonbusy disk;
   b. read the old data sector and the selected parity sector;
   c. compute the new parity sector according to (7);
   d. write the new data sector and the new parity sector to their corresponding disks;
   e. read back the parity and data sectors for verification; and
   f. signal the host for "write" completion.

3 Average Delay

3.1 Analysis

Fig. 4a shows a queuing model for DMP Disk Array. Requests sent from host become *jobs* to be served in the servers. Job arrivals are assumed to be a Poisson process with rate λ. A job is of the "write" type with probability α and of the "read" type with the remaining probability. Jobs not yet served by the disk array are queued in an FCFS queue. We call the job which is at the top of the queue the *Head Of Line (HOL) job*. The probability that the HOL job needs to access a particular disk is assumed to be the same for all disks. This assumption is usually not true, but it can be made true by distributing the frequently accessed data uniformly across all the disks. For mathematical convenience, we assume that the service time for a job is exponentially distributed with the service rates for a "write" job and a "read" job denoting as μ_w and μ_r, respectively.

Let random variables N_w and N_r denote the number of "write" jobs and "read" jobs in the disk array and N_q be the number of queuing jobs (including the HOL job) at any time. It is easy to see that the triplet (N_w, N_r, N_q) completely specifies the state of the system. Let $S_{w,r,q}$ denote the state of the system when $N_w = w$, $N_r = r$, and $N_q = q$. State transition will take place when a new job arrives or when a job in the system departs. Since the time spent in a state is exponentially distributed, the evolution of (N_w, N_r, N_q) is a continuous time Markov process. We define the transition probabilities to be

$$\begin{aligned} &P\left[S_{w',r',q'} \mid S_{w,r,q}\right] \\ &= P\left[\left(N_w, N_r, N_q\right) = (w', r', q') \text{ after state transition} \right. \\ &\quad \left. \mid \left(N_w, N_r, N_q\right) = (w, r, q) \text{ before state transition}\right]. \end{aligned} \tag{11}$$

Consider a particular state transition at time *t*. Define events E$_1$, E$_2$, and E$_3$ as:

E$_1$: A new job arrives at time *t*.
E$_2$: A "read" job departs at time *t*.
E$_3$: A "write" job departs at time *t*.

The event E$_1$ is listed in the first column of Table 1. A complete table listing the other two events is given in [3]. The disk array is at $S_{w,r,q}$ immediately before *t*, i.e., at time $t - \delta t$ ($\delta t \rightarrow 0$). The probability that a new job will arrive in the interval $(t - \delta t, t)$ is $\lambda \delta t$ if $\delta t \rightarrow 0$. Similarly, the probabilities that the disk array will finish serving a "write" job and a "read" job in this small time interval are $\mu_w \delta t$ and $\mu_r \delta t$, respectively. Therefore,

$$\begin{aligned} P[E_1] &= \frac{\lambda}{\lambda + w\mu_w + r\mu_r} \\ P[E_2] &= \frac{r\mu_r}{\lambda + w\mu_w + r\mu_r} \\ P[E_3] &= \frac{w\mu_w}{\lambda + w\mu_w + r\mu_r}. \end{aligned} \tag{12}$$

When a new job arrives (i.e., E_1 occurs), the probabilities that this HOL job is of the "read" type and of the "write" type are $1 - \alpha$ and α, respectively. However, a different probability for each of the job types for the HOL job is found when a job departs (i.e., either E_2 or E_3 occurs). We denote the probabilities that the HOL job is of the "write" type and of the "read" type by h_w and $1 - h_w$, respectively, for system transitions due to job departures. Probability h_w can be derived as follows: At the previous state change, the HOL job was blocked because it requires the access of one or more busy disks. By that time there were $2w + r$ busy disks. If the HOL job is of the "read" type, the probability of blocking k_r is:

$$k_r = \frac{\text{number of busy disks}}{\text{total number of disks}} = \frac{2w + r}{M}. \tag{13}$$

On the other hand, if the HOL job is of the "write" type, the probability of blocking k_w can be derived as follows: We shall call the disk which the HOL job targets for data modification the *data disk* and the R disks storing the required parity information the *parity disks*. Let events ξ_1 and ξ_2 be:

ξ_1: The data disk to be accessed was free at the previous state change.

ξ_2: At least one of the *R* parity disks was free at the previous state change.

Then,

$$\begin{aligned} 1 - k_w &= P[\text{no blocking}] \\ &= P[\xi_1 \ and \ \xi_2] \\ &= P[\xi_1] P[\xi_2 \mid \xi_1]. \end{aligned} \tag{14}$$

$P[\xi_2 | \xi_1]$ is given by:

$$\begin{aligned} P[\xi_2 \mid \xi_1] &= 1 - \frac{\begin{pmatrix} \text{number of ways to} \\ \text{choose } R \text{ parity disks} \\ \text{from } 2w + r \text{ busy disks} \end{pmatrix}}{\begin{pmatrix} \text{number of ways to} \\ \text{choose R parity disks} \\ \text{from } M - 1 \text{ disks} \end{pmatrix}} \\ &= \begin{cases} 1 & 2w + r < R \\ 1 - \dfrac{\binom{2w+r}{R}}{\binom{M-1}{R}} & 2w + r \geq R. \end{cases} \end{aligned} \tag{15}$$

Substituting into (14) and solving for k_w we obtain

$$k_w = \begin{cases} \dfrac{2w + r}{M} & 2w + r < R \\ \dfrac{2w + r}{M} + \dfrac{M - (2w + r)}{M} \dfrac{\binom{2w+r}{R}}{\binom{M-1}{R}} & 2w + r \geq R. \end{cases} \tag{16}$$

Having obtained k_r and k_w, h_w is given by:

$$h_w = \frac{k_w}{k_r + k_w}. \tag{17}$$

Conditioning on event E_i and giving the type of the HOL job, the probabilities of having different numbers of jobs located in the queue and in the disk array are listed in the sixth column of Table 1. For example, the first row of Table 1 corresponds to the case that a "read" job enters an empty queue is blocked or the number of "read" jobs and "write" jobs in the disk array remain the same and q' becomes 1. The probability of having the new triplet $(w', r', q') = (w, r, 1)$ after t under the two given conditions is denoted as a_1. Similarly the fifth row of Table 1 corresponds to the case that a "write" job enters an empty queue and gets served immediately. The probability of having $(w', r', q') = (w + 1, r, 0)$ after t under

TABLE 1
Transition Probabilities of Event E_1 for Average Delay Analysis

E_i	New states (w',r',q')			Type of the H.O.L job	P[w',r',q' | E_i]	Type of the new HOL job	P[$S_{w,r,q}$, $S_{w',r',q'}$ | E_i]
E_1 {a new job arrives}	q=0	q'=1	w'=w, r'=r	read (1-α)	a_1	-	(1-α)a_1
				write (α)	a_2	-	αa_2
			all other new states	-	0	-	0
		q'=0	w'=w, r'=r+1	read (1-α)	a_3	-	(1-α)a_3
			w'=w+1, r'=r	write (α)	a_4	-	αa_4
			all other new states	-	0	-	0
		q'>1	all new states	-	0	-	0
	q≥1	q'=q+1	w'=w, r'=r	-	1	-	1
			all other new states	-	0	-	0
		q'≠q+1	all new states	-	0	-	0

A complete table showing the transition probabilities of all other events can be found in [3].

the two given conditions is denoted by a_3, as shown. The derivation of all a_is is given in [3].

The seventh column of Table 1 shows the type of the new HOL job and its corresponding probability. A new HOL job is blocked at time t with probabilities $k'r$ and $k'w$ when it is of the "read" type and of the "write" type, respectively. Similarly to the derivations on k_r and k_w, $k'r$ and $k'w$ can be obtained from (13) and (16) by changing the number of busy disks from $2w + r$ to $2w' + r'$.

The last column of Table 1 shows $P[S_{w',r',q'}|S_{w,r,q}, E_i]$. By removing the condition on E_i, the transition probabilities are thus obtained:

$$P\big[S_{w',r',q'} \mid S_{w,r,q}\big] = \sum_{i=1}^{3} P\big[S_{w',r',q'} \mid S_{w,r,q}, E_i\big] P[E_i]. \tag{18}$$

Having obtained the transition probabilities, the equilibrium distribution of different states can be computed in the usual way. The expected numbers of jobs in the queue are given by:

$$E[N_q] = \sum_{w=0}^{\lfloor M/2 \rfloor} \sum_{r=0}^{M} \sum_{q=0}^{\infty} qP[S_{w,r,q}]. \tag{19}$$

Finally, by Little's formula, the job's sojourn time D is given by

$$D = \frac{E[N_q]}{\lambda} + \left(\frac{\alpha}{\mu_w} + \frac{(1-\alpha)}{\mu_r}\right). \tag{20}$$

3.2 Numerical Example

As an example, consider a small DMP Disk Array with data storage capacity of four disks. We assume in this example that $\mu_w = 30$ jobs/sec and $\mu_r = 50$ jobs/sec. Figs. 5 and 6 show both the analytic and simulation results for this disk array and we observe that they match very well with each other. Note that, for all simulation results shown in this paper, we have extended the simulation time sufficiently long to make the 95 percent confidence intervals smaller than the size of the simulation points shown.

Fig. 5 shows the job delay against the arrival rate when half of the jobs are of the "write" type. As indicated by the curve, the maximum throughput for RAID level 5 is about 50 jobs/sec. When one parity disk is added to the disk array (i.e., when $M = 6$ and $R = 2$), we find that the average job delay is reduced under all traffic conditions and the maximum throughput is increased by about 13 percent when compared to RAID level 5. If one more parity disk is used ($R = 3$), we find that the job delay is further reduced under all traffic conditions and 23 percent increase in maximum throughput is observed.

Fig. 6 shows the job delay against the arrival rate when 20 percent of the jobs are of the "write" type ($\alpha = 0.2$). We observe from the figure that DMP Disk Array with $R = 2$ and $R = 3$ again performs significantly better than RAID level 5 under all traffic levels.

4 Maximum Throughput

4.1 Analysis

Although the analysis given in Section 3 provides an exact solution on average job delay, the computation is very demanding when the disk array is large. In the following, we present a simplified analysis which gives the maximum throughput for DMP Disk Array. A modified model, shown in Fig. 4b, is used in our analysis. Compared with the previous model (Fig. 4a), the FCFS queue is removed and

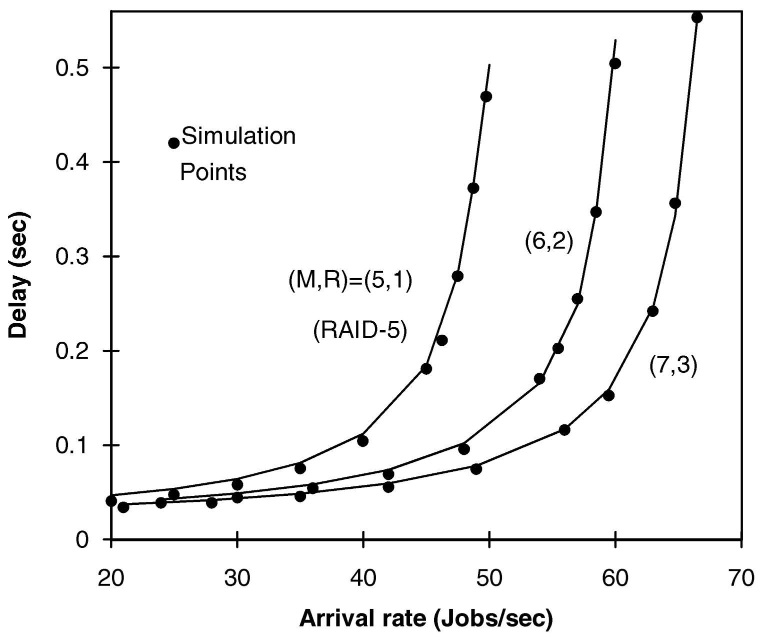

Fig. 5. Average job delay against the arrival rate for small DMP Disk Arrays (data storage capacity M-R is four disks). Half the jobs are of “write” type ($\alpha = 0.5$).

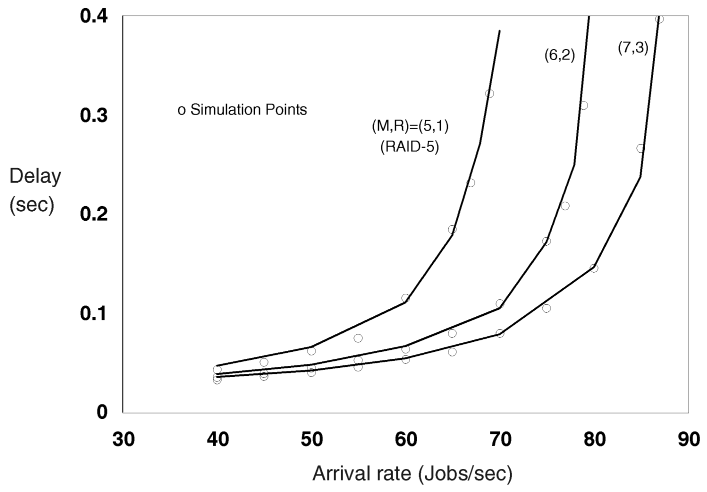

Fig. 6. Average job delay against the arrival rate for small DMP Disk Arrays (data storage capacity M-R is four disks). Eighty percent of the jobs are of the “read” type ($\alpha = 0.2$).

we assume that there is always a new job available at the input of the disk array. All other previously used assumptions are used in this maximum throughput analysis. Since the queue is removed, the system's state can solely be specified by N_w and N_r and is denoted by $S_{w,r}$. State transition will take place when a job in the disk array departs, i.e., either E_2 or E_3 occurs. Since the time spent in a state is exponentially distributed, the evolution of N_w and N_r remains a continuous time Markov process. As before, we define the transition probabilities to be

$$\begin{aligned}&P[S_{w,r}, S_{w',r'}] = \\ &P[N_w = w', N_r = r' \text{ after system transition} \mid \\ &\qquad N_w = w, N_r = r \text{ before system transition}].\end{aligned} \tag{21}$$

Consider a particular state transition occurs at time t. The probability of occurrence of E_2 and E_3 is given by

$$\begin{aligned}P[E_2] &= \frac{r\mu_r}{w\mu_w + r\mu_r} \\ P[E_3] &= \frac{w\mu_w}{w\mu_w + r\mu_r}.\end{aligned} \tag{22}$$

Given that a specific event E_i occurs, the transition probabilities $P[S_{w,r}, S_{w',r'}|E_i]$ are listed in the last column of Table 2.[3] The derivations on probabilities a_i shown in the table are given in [3]. Having obtained the transition probabilities, the equilibrium distribution of different states can be computed as before. The expected time between successive job departures X is given by

3. Table 2 shows only the transition probabilities of event E_1. A complete table for all events can be found in [16].

TABLE 2
Transition Probabilities of Event E_2 for Maximum Throughput Analysis

E_i	New States (w',r')	Type of the H.O.L job	P[w',r'|E_i]	Type of the new HOL job	P[$S_{w,r}$, $S_{w',r'}$|E_i]
E_2 {a read job departs}	w'=w, r'=r-1	read (1-h_w)	a_5	-	(1-h_w)a_5
		write (h_w)	a_6	-	$h_w a_6$
	w'=w, r'=r	read (1-h_w)	a_7	read ((1-α)k'_r)	(1-h_w)(1-α)$a_7 k'_r$
				write (αk'_w)	(1-h_w)α$a_7 k'_w$
	w'=w+1, r'=r-1	write (h_w)	a_8	read ((1-α)k'_r)	h_w(1-α)$a_8 k'_r$
				write (αk'_w)	h_wα$a_8 k'_w$
	w'=w, r'>r	read (1-h_w)	a_9	read ((1-α)k'_r)	(1-h_w)(1-α)$a_9 k'_r$
				write (αk'_w)	(1-h_w)α$a_9 k'_w$
	w'>w+1, r'=r-1	write (h_w)	a_{10}	read ((1-α)k'_r)	h_w(1-α)$a_{10} k'_r$
				write (αk'_w)	h_wα$a_{10} k'_w$
	w'≥w+1, r'≥r	read (1-h_w)	a_{11}	read ((1-α)k'_r)	(1-h_w)(1-α)$a_{11} k'_r$
				write (αk'_w)	(1-h_w)α$a_{11} k'_w$
		write (h_w)	a_{12}	read ((1-α)k'_r)	h_w(1-α)$a_{12} k'_r$
				write (αk'_w)	h_wα$a_{12} k'_w$
	all other new states	-	0	-	0

A complete table showing the transition probabilities of all other events can be found in [3].

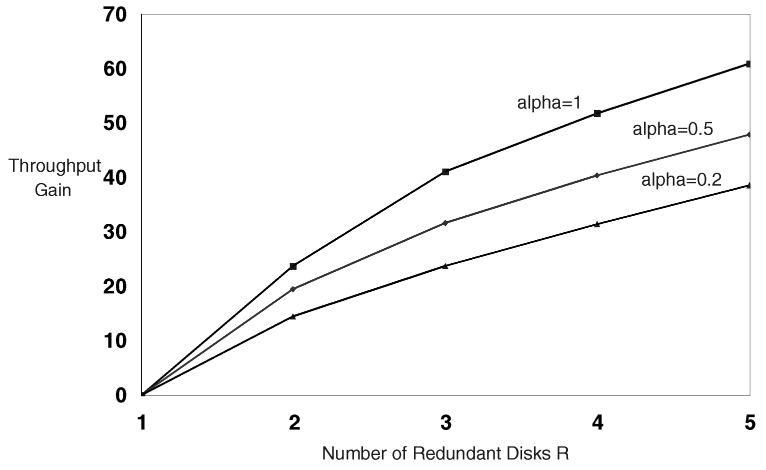

Fig. 7. Throughput gain over that of RAID level 5 as a function of R for small DMP Disk Arrays (data storage capacity M-R is four disks).

$$X = \sum_{w=0}^{\lfloor M/2 \rfloor} \sum_{r=0}^{M} P[S_{w,r}] \frac{1}{w\mu_w + r\mu_r}. \tag{23}$$

Finally, the maximum throughput for DMP Disk Arrays T is

$$T = \frac{1}{X}. \tag{24}$$

4.2 Numerical Examples

Fig. 7 shows the throughput gain over that of RAID level 5 against R for a small DMP Disk Array with data storage capacity of four disks. We observe from the figure that increasing the number of parity disks R will always increase the maximum throughput for DMP Disk Array. When the proportion of "write" jobs is higher, the increase in

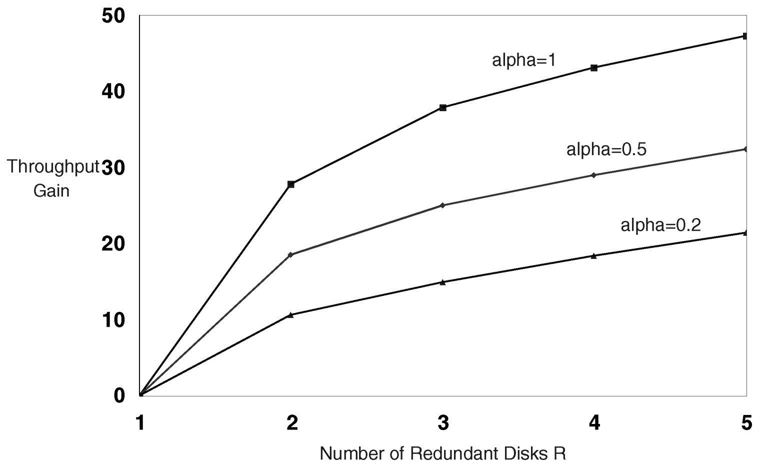

Fig. 8. Throughput gain over that of RAID level 5 as a function of R for large DMP Disk Arrays (data storage capacity M-R is 13 disks).

maximum throughput is more apparent because DMP Disk Array will reduce the queuing time of "write" jobs. Considering the case of all "write" jobs ($\alpha = 1$), DMP Disk Array with $R = 2$ provides 24 percent increase in maximum throughput when compared to RAID level 5. Further increasing R to three provides an additional 17 percent increase in maximum throughput. These performance figures match well with the numerical examples given in the previous section. When R is greater than three, linear increase in maximum throughput is observed for each parity disk added. From the figure, we also observe that significant throughput gain is obtained for a typical 20 percent proportion of "write" jobs.

Fig. 8 plots the throughput gain over that of RAID level 5 against R for a large DMP Disk Array with data storage capacity of 13 disks. When compared with Fig. 7, we observe that DMP Disk Array with $R = 2$ provides even more notable increase in maximum throughput than the previous case. We can thus conclude that a DMP Disk Array with $R = 2$ provides the best cost/performance ratio.

5 Simulation with Precise Disk Model

In our previous analysis, disk service time is assumed to be exponentially distributed. This is usually not true for practical disk drives. To better understand the performance of DMP Disk Array in practice, we perform simulation on DMP Disk Array with a precise disk model. In our simulation, disks are *not* assumed to be rotationally synchronized and their simulation parameters are summarized in Table 3. Each disk access involves a seek time, a latency, and a data transfer time [16]. We use the seek profile in [17], which states that the seek time *Tseek* (in mSec) is related to seek distance x (in number of cylinders) by:

$$T_{seek} = \begin{cases} 0 & \text{for } x = 0 \\ 0.4623\sqrt{x-1} + 0.0092(x-1) + 2 & \text{for } x > 0. \end{cases} \tag{25}$$

Latency is assumed to be uniformly distributed. Data transfer time for one sector is equal to the disk revolution time divided by the number of sectors per track, as given in Table 3. With that, the mean service time for "write" jobs is computed to be 33.3 ms and, for "read" jobs, it is 20 ms. The corresponding service rates are therefore the same as those in the previous examples. As stated in [18], this kind of disk modeling provides more than 94 percent accuracy when ignoring the disk caching effect. Since disk caching has little impact on "write" performance (which we are most

TABLE 3
Disk Parameters Used in Simulation

Cylinders per disk	1024
Tracks per cylinder	14
Sectors per track	48
Bytes per sector	512
Revolution time	13.3 ms
Single cylinder seek time	2 ms
Average seek time	13 ms
Max. data transfer rate	1.7 MB/s

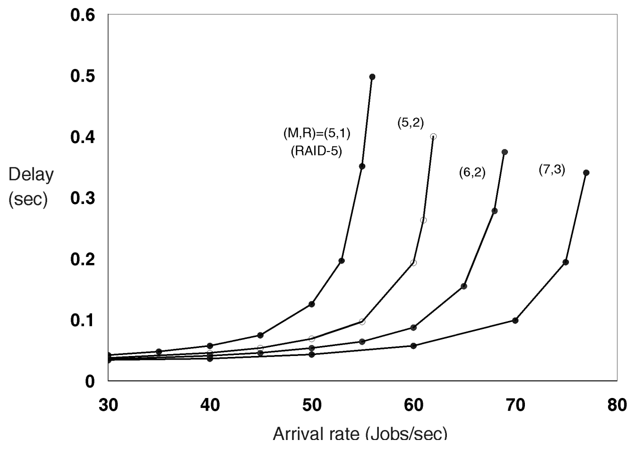

Fig. 9. Simulation results on average job delay against arrival rate for small disk arrays (data storage capacity M-R is four disks). Half the jobs are "write" ($\alpha = 0.5$).

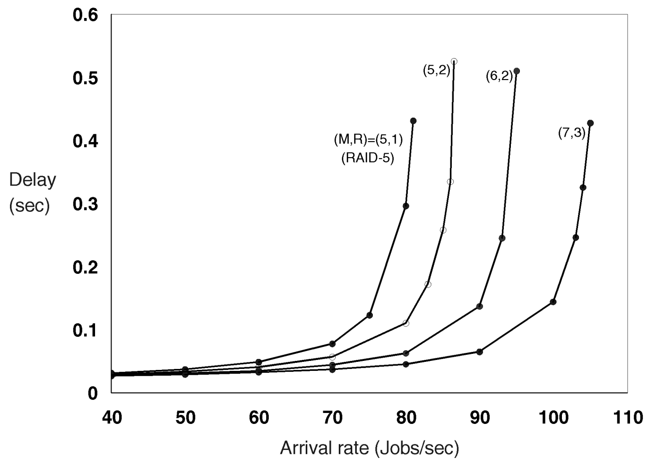

Fig. 10. Simulation results on average job delay against arrival rate for small disk arrays (data storage capacity M-R is four disks). Eighty percent of the jobs are of the "read" type ($\alpha = 0.2$).

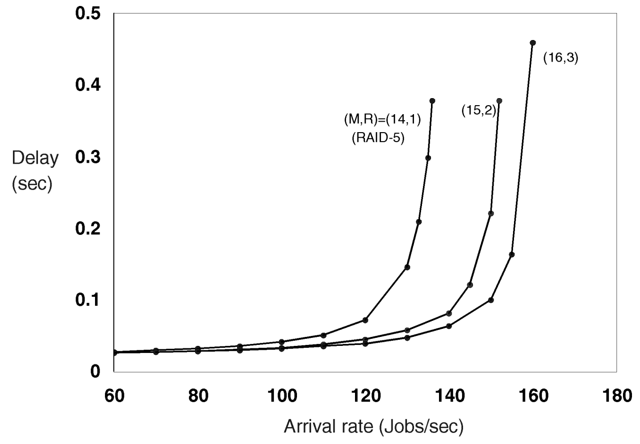

Fig. 12. Simulation results on average job delay against arrival rate for large disk arrays (data storage capacity M-R is 13 disks). Eighty percent of the jobs are of the "read" type ($\alpha = 0.2$).

interested in), we can thus assume that the system has no disk caching mechanism.

Figs. 9, 10, 11, and 12 show the simulation results with the precise disk model. We first consider a small disk array with data storage capacity of four disks and half the jobs are of the "write" type. Fig. 9 shows that the maximum throughput for RAID level 5 is about 56 jobs/sec. If DMP Disk Array with $M = 6$ and $R = 2$ is used, the average job delay is reduced under all traffic conditions and the maximum throughput is increased by 23 percent as compared to RAID level 5. The maximum throughput is further increasted by about 40 percent when DMP Disk Array with $M = 7$ and $R = 3$ is used. We also observe that when the number of disks M is fixed, DMP Disk Array with $R = 2$ still provides 11 percent improvement in maximum throughput.

Fig. 10 shows the results for $\alpha = 0.2$. We observe that DMP Disk Array with $R = 2$ and $R = 3$ provides 17 percent and 29 percent increases in I/O throughput, respectively. The increase in throughput when $M = 5$ and $R = 2$ is about 7 percent.

Figs. 11 and 12 show the delay throughput characteristics of a typical large disk array with data storage capacity of 13 disks. DMP Disk Array again provides significant I/O throughput increase. When half the jobs are of the "write" type (Fig. 11), the increase on maximum throughput for DMP Disk Array with $R = 2$ is about 21 percent, whereas the corresponding increase in system cost is at most 7 percent. Under the condition of 20 percent of the jobs are of the "write" type (Fig. 12), we find that the 7 percent increase in system cost can still give 12 percent higher throughput.

Figs. 13 and 14 show the case when the number of disks, M, is fixed at 14 disks. When half the jobs are of the "write" type (Fig. 13), the increase on maximum throughput for DMP Disk Array with $R = 2$ is about 17 percent. Note that there is *no* increase in system cost. Under the condition of 20 percent of the jobs are of the "write" type (Fig. 14), we find that DMP Disk Array with $R = 2$ still provides 8 percent higher throughput at no additional system cost.

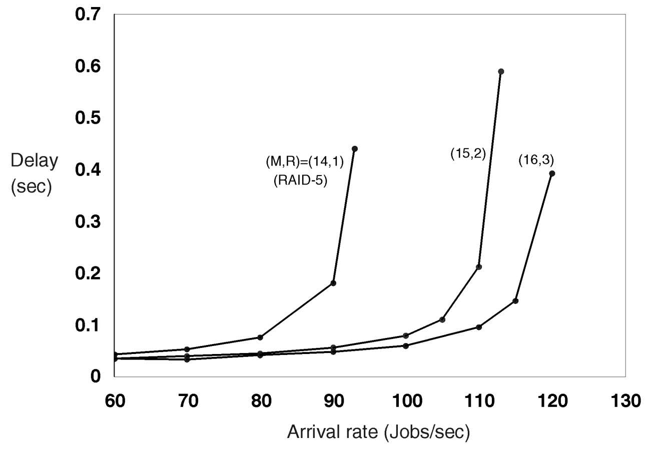

Fig. 11. Simulation results on average job delay against arrival rate for large disk arrays (data storage capacity M-R is 13 disks). Half the jobs are of "write" type ($\alpha = 0.5$).

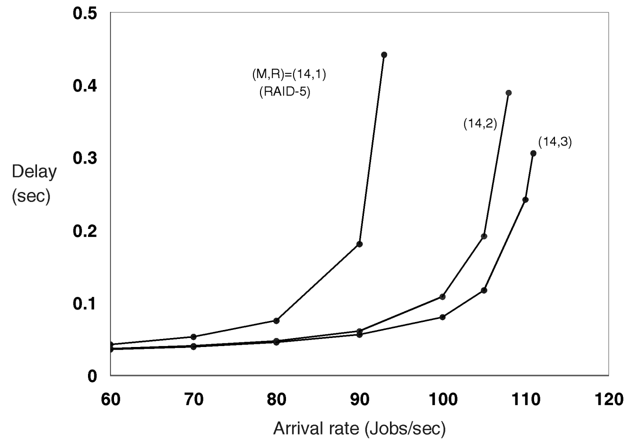

Fig. 13. Simulation results on average job delay against arrival rate for large disk arrays (total number of disks, M, is 14). Half the jobs are of "write" type ($\alpha = 0.5$).

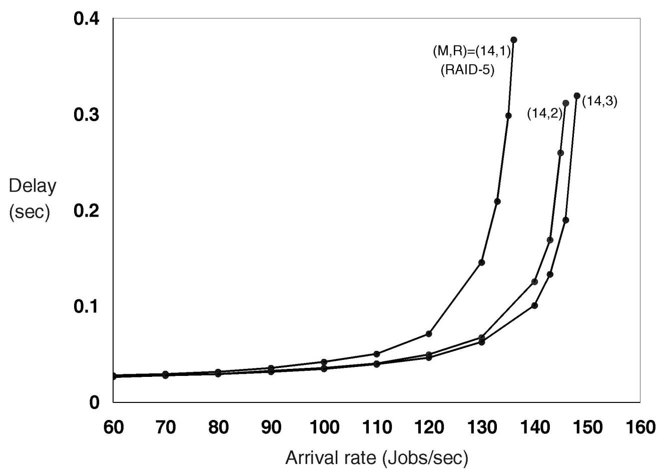

Fig. 14. Simulation results on average job delay against arrival rate for large disk arrays (total number of disks, M, is 14). Eighty percent of the jobs are of "read" type ($\alpha = 0.2$).

6 Conclusions

In this paper, we propose a new RAID architecture, called Dynamic Multiple Parity (DMP) Disk Array, for fast database system applications. The DMP Disk Array provides significant improvement on I/O throughput over the RAID level 5. The DMP Disk Array also inherently has the sparing property so that it has a higher survivability under disk failure conditions. Delay and maximum throughput analysis on DMP Disk Array is performed. Simulation with precise disk model shows that, for a typical "write" job proportion of 20 percent, DMP Disk Array can provide nearly 20 percent improvement on I/O performance over that of RAID level 5 when one extra parity disk is used.

References

[1] S.Y.W. Su, *Database Computers.* McGraw-Hill, 1988.
[2] T. Bowen, G. Gopal, G. Herman, and W. Mansfield, "A Scale Database Architecture for Network Services," *IEEE Comm. Magazine,* vol. 29, no. 1, Jan. 1991.
[3] K.H. Yeung, "High Performance Disk Array Architectures," PhD dissertation, The Chinese Univ. of Hong Kong, 1995.
[4] J.L. Encarnacao and P.C. Lockemann, *Engineering Databases.* Springer-Verlag, 1990.
[5] K.H. Yeung and T.S. Yum, "Dynamic Parity Logging Disk Arrays for Engineering Database Systems," *IEE Proc.—Computers and Digital Techniques,* vol. 144, no. 5, pp. 255-260, Sept. 1997.
[6] R.H. Katz, *Information Management for Engineering Design,* Springer-Verlag, 1985.
[7] M. Singhal and N.G. Shivaratri, *Advanced Concepts in Operating Systems,* chapter 20. McGraw-Hill, 1994.
[8] F.R. McFadden, J.A. Hoffer, and M.B. Prescott, *Modern Database Management,* fifth ed. Addison-Wesley, 1999.
[9] S. Ceri and G. Pelagatti, *Distributed Databases: Principles and Systems,* chapter 8. McGraw-Hill, 1984.
[10] E.K. Lee and R.H. Katz, "The Performance of Parity Placements in Disk Arrays," *IEEE Trans. Computers,* vol. 42, no. 6, June 1993.
[11] J. Menon and D. Mattson, "Comparison of Sparing Alternatives for Disk Array," *Proc. 19th Int'l Symp. Computer Architecture,* pp. 318-329, May 1992.
[12] M.Y. Kim, "Synchronized Disk Interleaving," *IEEE Trans. Computers,* vol. 35, no. 11, pp. 978-988, Nov. 1986.
[13] N.C. Wilhelm, "A General Model for the Performance of Disk Systems," *J. ACM,* vol. 24, no. 1, pp. 14-31, Jan. 1977.
[14] S.W. Ng, "Improving Disk Performance via Latency Reduction," *IEEE Trans. Computers,* vol. 40, no. 1, pp. 22-30, Jan. 1991.
[15] A.L. Narasimha Reddy and P. Banerjee, "An Evaluation of Multiple-Disk I/O Systems," *IEEE Trans. Computers,* vol. 38, no. 12, Dec. 1989.
[16] S.W. Ng, "Advances in Disk Technology: Performance Issues," *Computer,* vol. 31, no. 5, pp. 75-81, May 1998.
[17] E.K. Lee and R.H. Katz, "The Performance of Parity Placements in Disk Arrays," *IEEE Trans. Computers,* vol. 42, no. 6, pp. 651-664, June 1993.
[18] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling," *Computer,* vol. 27, no. 3, pp. 17-28, Mar. 1994.

K.H. Yeung received the BSc degree from the Chinese University of Hong Kong in 1984, the postgraduate diploma and MSc degree from the City University of Hong Kong in 1989 and 1991, respectively, and the PhD degree from the Chinese University of Hong Kong in 1995. He was an R&D engineer with a PC manufacturer for one year before joining the City University of Hong Kong in 1985. He was an analyst programmer for five years before transferring to the Department of Electronic Engineering in 1990, where he is currently an associate professor. His research interests include Internet system design, mobile communication, information delivery, and VoD architectures. Besides research, he is also active in providing consultancy services to local industry in the areas of computer networking, communication systems, and Internet systems. He is also a Certified Cisco Academy Instructor. He is a member of the IEEE.

T.S. Yum is a graduate of Columbia University. He worked at Bell Telephone Laboratories for two and a half years and taught at National Chiao Tung University, Taiwan, for two years before joining The Chinese University of Hong Kong in 1982. He has published original research on packet switched networks with contributions in routing algorithms, buffer management, deadlock detection algorithms, message resequencing analysis, and multiaccess protocols. In recent years, he branched out to work on the design and analysis of cellular network, lightwave networks, and video distribution networks. He believes that the next challenge is designing an intelligent network that can accommodate the needs of individual customers. Professor Yum's research benefits a lot from his graduate students. Seven of them are now professors at local and overseas universities. He is on the editorial board of six journals on communications and information science, including the recently launched *IEEE Transactions on Multimedia*. He is a senior member of the IEEE.